TÓM LƯỢC BÀI GIẢNG

(Vũ Quốc Hoàng)

KIỂU MẢNG ĐỘNG vector (C++ STL)

Chủ đề

- Kiểu mảng động `vector` (C++ STL)

Tài liệu

[1] Tony Gaddis, *Starting out with C++ From Control Structures through Objects*, Pearson, 8th edition, 2015.

[2] Vũ Quốc Hoàng, *Bí kíp luyện Lập trình C (Quyển 1)*, hBook, 2017.

Đọc tài liệu

- Kiểu mảng động `vector` (C++ STL): Section 7.12 [1]

Kiến thức

- Đoạn mã C++ minh họa việc tạo và thao tác mảng động các số nguyên với kiểu `vector`

```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;

int main()
{
    cout << "Mang dong dung vector (C++ STL)." << endl;
    vector<int> vec;

    while(1)
    {
        int cmd, x;
        cout << "\nChon lenh (0-dung, 1-in, 2/3-them dau/cuoi,";
        cout << " 4/5-xoa dau/cuoi, 6-sap xep): ";
        cin >> cmd;

        if(cmd == 0)
            break;
        else if(cmd == 1)
        {
            cout << "Cac phan tu: ";
            for(int i = 0; i < vec.size(); i++)
                cout << vec[i] << " ";
            cout << endl;
```

```
        }
        else if(cmd == 2)
        {
            cout << "Nhap phan tu them vao dau: ";
            cin >> x;
            vec.insert(vec.begin(), x);
        }
        else if(cmd == 3)
        {
            cout << "Nhap phan tu them vao cuoi: ";
            cin >> x;
            vec.push_back(x);
        }
        else if(cmd == 4 && !vec.empty())
        {
            x = vec.front();
            vec.erase(vec.begin());
            cout << "Da xoa phan tu dau: " << x << endl;
        }
        else if(cmd == 5 && !vec.empty())
        {
            x = vec.back();
            vec.pop_back();
            cout << "Da xoa phan tu cuoi: " << x << endl;
        }
        else if(cmd == 6)
        {
            std::sort(vec.begin(), vec.end());
            cout << "Da sap xep (tang dan)" << endl;
        }
    }
}
```

- *Thư viện khuôn mẫu chuẩn* C++ (Standard Template Library, STL) cung cấp kiểu dữ liệu `vector` để tạo, quản lý và thao tác với mảng (cấp phát) động rất tiện lợi. Kiểu này cũng nằm trong không gian tên `std` và được `#include` từ file tiêu đề `vector`. Kiểu `vector` là một *kiểu chứa dãy* (sequence container) giúp quản lý dãy dữ liệu cùng kiểu, có thứ tự. Về mặt kĩ thuật, kiểu `vector` dùng mảng cấp phát động.
- Về mặt thuật ngữ, kiểu `vector` là một *khuôn mẫu lớp* (class template). Tùy theo kiểu (chung) Type của các phần tử mà ta có các kiểu vector cụ thể là `vector<Type>`. Chẳng hạn, trong ví dụ minh họa trên, ta đã dùng kiểu `vector<int>` cho mảng động các số nguyên `int`. Tương tự, ta có thể dùng kiểu `vector<double>` cho mảng động các số thực `double`, …
- Kiểu `vector` cung cấp các phương thức như: `size` cho biết số lượng phần tử, `empty` cho biết mảng có rỗng (không chứa phần tử nào) không, `insert` thêm phần tử vào vị trí nào đó, `erase` xóa phần tử ở vị trí nào đó, `push_back` thêm phần tử vào cuối, `pop_back` xóa phần tử cuối, …

- Kiểu `vector` cũng cung cấp toán tử truy cập phần tử (`[]`) giúp truy cập phần tử với chỉ số như trong mảng. Kiểu `vector` cũng hỗ trợ việc truy cập phần tử qua các phương thức: `at` truy cập phần tử tại vị trí nào đó, `front` truy cập phần tử đầu, `back` truy cập phần tử cuối. Kết quả trả về của các phương thức này là *tham chiếu* (C++ reference) đến các phần tử tương ứng. Đặc biệt, phương thức `at` có <u>hỗ trợ việc kiểm tra truy cập quá chỉ số mảng</u>.
- Việc truy cập các phần tử trong `vector` cũng có thể được thực hiện qua *đối tượng duyệt* (iterator). Phương thức `begin` trả về đối tượng duyệt ứng với phần tử đầu, end trả về đối tượng duyệt <u>sau phần tử cuối</u>. Có thể hình dung các đối tượng duyệt này như là con trỏ trỏ đến phần tử tương ứng, mà từ đó ta có thể truy cập được với *toán tử giải tham chiếu* (`*`). Chẳng hạn trong ví dụ minh họa trên ta có thể thay đoạn code `x = vec.front();` bằng đoạn code `x = *vec.begin();` hay đoạn code `x = vec.back();` bằng đoạn code `x = *(vec.end() - 1);`
- Thư viện STL cũng cung cấp các *thuật toán* (algorithm) hay dùng trên các kiểu chứa. Chẳng hạn hàm `sort` giúp sắp xếp các phần tử trong một `vector` (và nhiều kiểu chứa khác). Hàm này dùng các đối tượng duyệt để truy cập đến các phần tử. Hàm này cũng nằm trong không gian tên `std` và được `#include` từ file tiêu đề `algorithm`. Về mặt thuật ngữ, hàm `sort` là một *khuôn mẫu hàm* (function template) vì tùy theo kiểu của các đối tượng duyệt mà ta có các hàm `sort` cụ thể.

Kĩ năng

- Thành thạo việc dùng kiểu `vector` để quản lý mảng động
- Quen thuộc việc dùng thư viện STL

Lưu ý

- Thư viện STL cung cấp nhiều *"cấu trúc dữ liệu"* (data structure) và "*thuật toán*" (algorithm) rất hay dùng nên sinh viên cần thành thạo việc sử dụng thư viện này

Bài tập

1. Tìm một nguồn (web) tin cậy và tra cứu kiểu `vector` (mô tả kiểu, các toán tử, các phương thức, …), hàm `sort` và các "thuật toán" khác trong thư viện STL C++.
2. Làm lại các bài tập trong Bài 4.1, 4.2 và 4.3 [2] bằng kiểu `vector`. *<u>Gợi ý</u>*: Các phần tử của `vector` có thể có kiểu bất kì (miễn là cùng kiểu) nên chúng có thể là các `vector`. Như vậy ta có thể dùng kiểu `vector< vector<int> >` cho mảng động 2 chiều các số nguyên.